

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1



**BÁO CÁO
KIẾN TRÚC VÀ THIẾT KẾ
PHẦN MỀM**

Họ và tên: Đỗ Duy Kiên

Mã sinh viên: B20DCCN352

Lớp: D20CNPM02

Nhóm bài tập: 05

Hà Nội – 2024

Câu 1: Mô tả bằng dạng Bảng và ngôn ngữ tự nhiên các chức năng tương ứng với các actor và phi chức năng của Hệ thống ecomSys

❖ Mô tả hệ thống bằng ngôn ngữ tự nhiên.

- Hệ thống ecomSys là một nền tảng trực tuyến cho phép người dùng mua và bán hàng hóa dịch vụ qua internet. Hệ thống cung cấp một giao diện dễ sử dụng cho người dùng để tìm kiếm các sản phẩm theo nhu cầu. Người dùng có thể truy cập vào hệ thống từ bất kỳ thiết bị kết nối internet nào. Mỗi sản phẩm trên hệ thống được hiển thị với thông tin chi tiết bao gồm hình ảnh, mô tả, giá cả và thông tin giao hàng. Khách hàng có thể đăng ký tài khoản, đăng nhập, xem thông tin sản phẩm, thêm sản phẩm vào giỏ hàng và tiến hành thanh toán trực tuyến hoặc thanh toán khi nhận hàng. Khách hàng cũng có thể tìm kiếm sản phẩm theo tên sản phẩm, thể loại,.. Đối nhân viên hệ thống cũng cung cấp các công cụ quản lý sản phẩm, quản lý đơn hàng, xét duyệt đơn hàng. Quản trị viên có thể

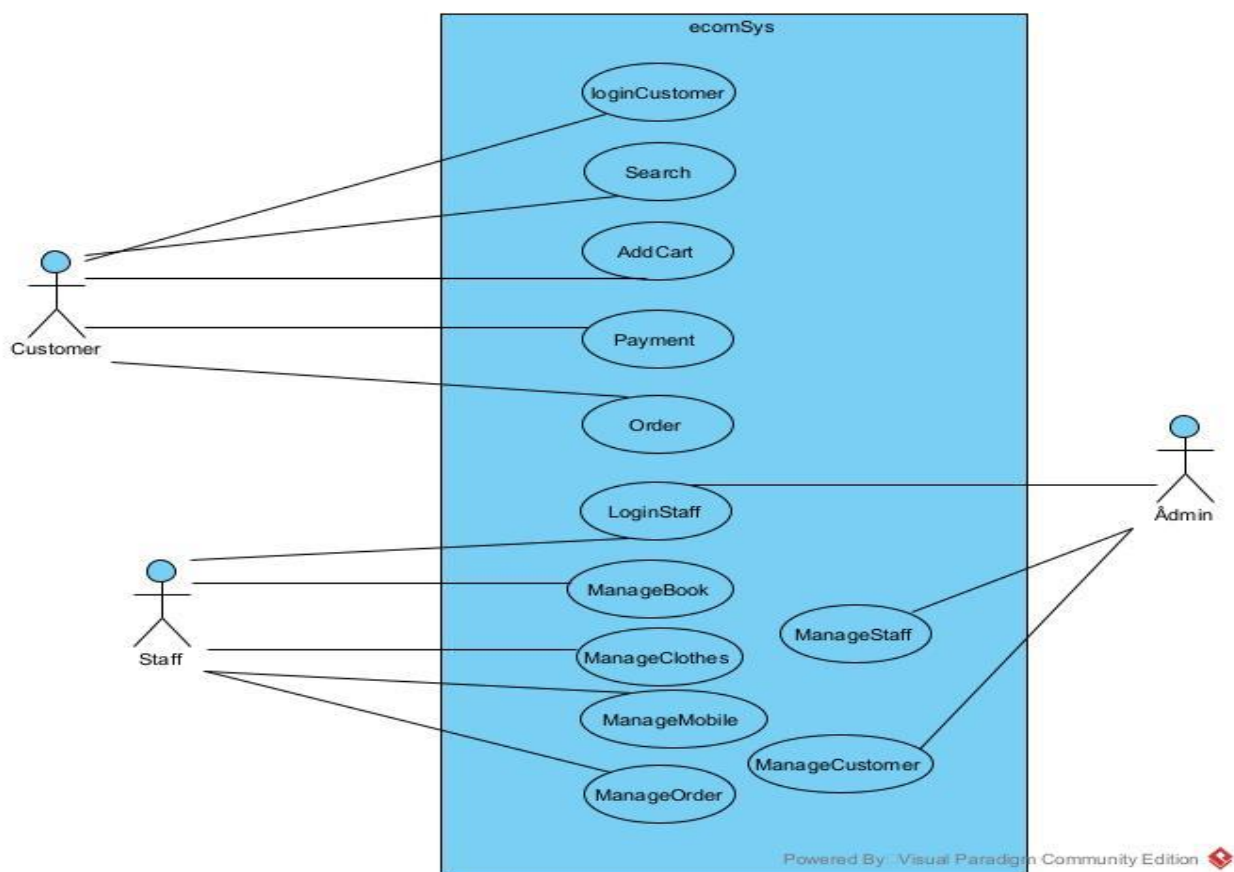
❖ Bảng mô tả các chức năng.

Actor	Chức năng	Mô tả
Customer	Add to cart	Khách hàng click vào thêm giỏ hàng ở giao diện chi tiết sản phẩm, sản phẩm sẽ được thêm vào giỏ hàng. Những sản phẩm ở trong giỏ hàng được coi là sản phẩm bạn muốn mua.
	Search	Khách hàng có thể tìm kiếm sản phẩm theo keyword, hình ảnh, âm thanh, Hệ thống sẽ đưa ra các sản phẩm khách hàng muốn thấy. Khách hàng cũng có thể xem thông tin sản phẩm <ul style="list-style-type: none">- Tìm kiếm bằng keyword: Khách hàng nhập từ khóa là mã hoặc tên sản phẩm mà mình muốn tìm vào thanh tìm kiếm.- Tìm kiếm bằng âm thanh: Người sử dụng nói mã, tên sản phẩm muốn tìm
	Payment	Khách hàng chọn phương thức thanh toán cho sản phẩm. Có 2 phương thức thanh toán là thanh toán trực tuyến và thanh toán khi nhận hàng <ul style="list-style-type: none">- Thanh toán trực tuyến: Người dùng sử dụng các ví điện tử, thẻ ngân hàng, visa để thanh toán- Thanh toán khi nhận hàng: Khi sản phẩm đến tay người dùng, người dùng sẽ thanh toán với người giao hàng

	Order	Sau khi khách hàng chọn phương thức thanh toán xong đơn hàng sẽ được tạo và đang trong trạng thái chờ duyệt. Đơn hàng sẽ có 3 trạng thái là chờ duyệt, đang xử lý, đã giao.
Staff	Manage Book	Nhân viên có thể thêm, sửa, xóa thông tin sách. Nhân viên có thể thêm
	Manage Clothes	Nhân viên có thể thêm, sửa, xóa thông tin quần áo. Nhân viên có thể thêm
	Manage Mobile	Nhân viên có thể thêm, sửa, xóa thông tin điện thoại. Nhân viên có thể thêm
	Manage Order	Nhân viên có thể xem tất cả các đơn hàng trên hệ thống và duyệt đơn hàng
Admin	Manage Customer	Quản trị viên hệ thống có thể quản lý tài khoản và thông tin khách hàng
	Manage Staff	Quản trị viên hệ thống có thể quản lý tài khoản và thông tin nhân viên

Câu 2: Vẽ Biểu đồ use case tổng quát và biểu đồ usecase chi tiết cho từng chức năng/dịch vụ

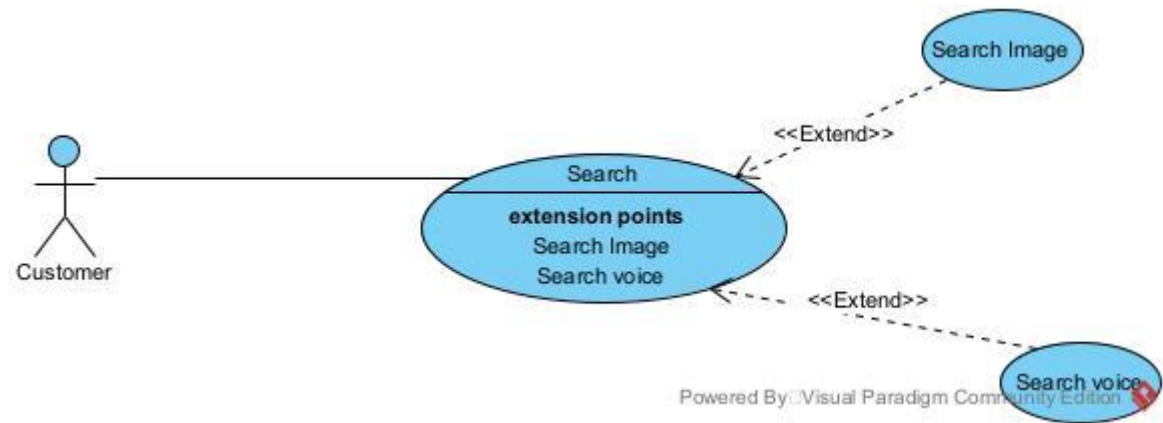
Sơ đồ usecase tổng quan:



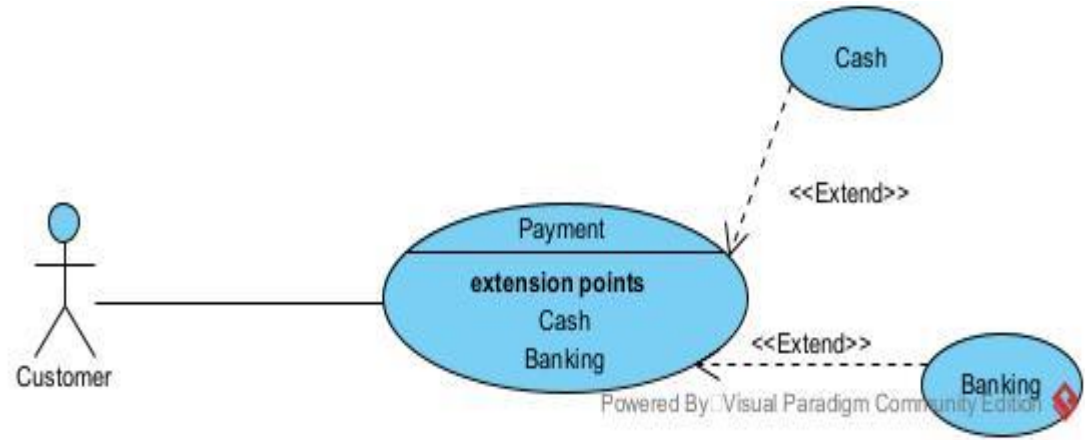
Biểu đồ usecase chi tiết customerLogin:



Biểu đồ usecase chi tiết search



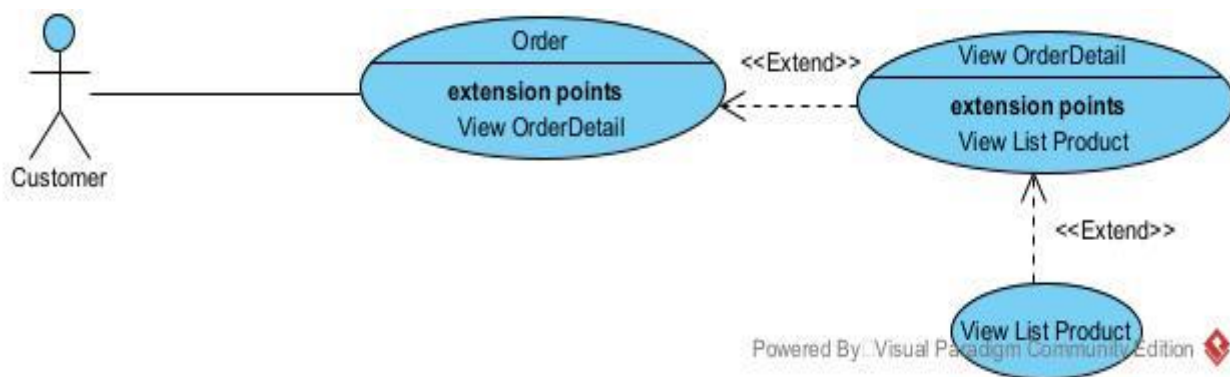
Biểu đồ usecase chi tiết payment



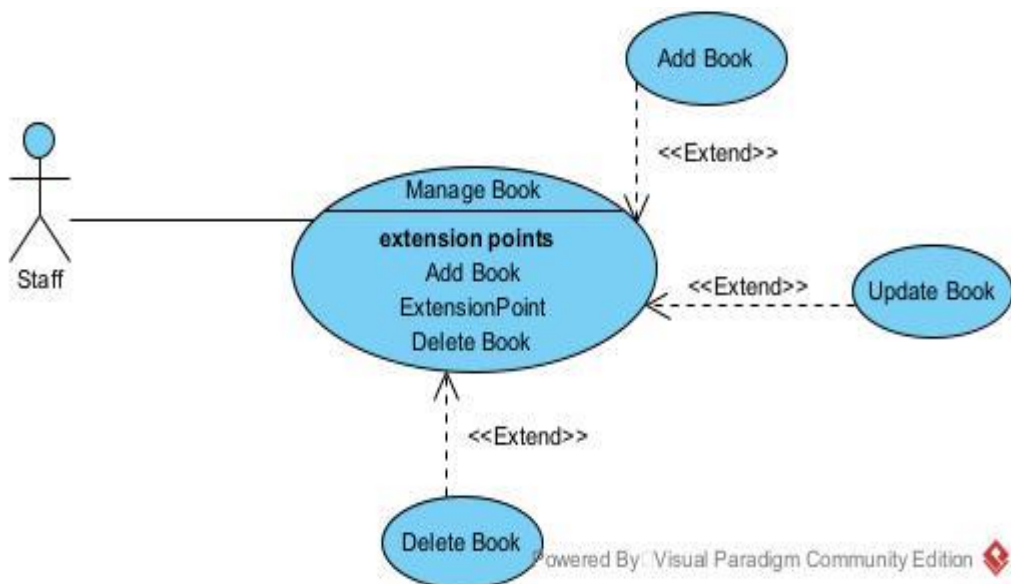
Biểu đồ usecase thêm vào giỏ hàng



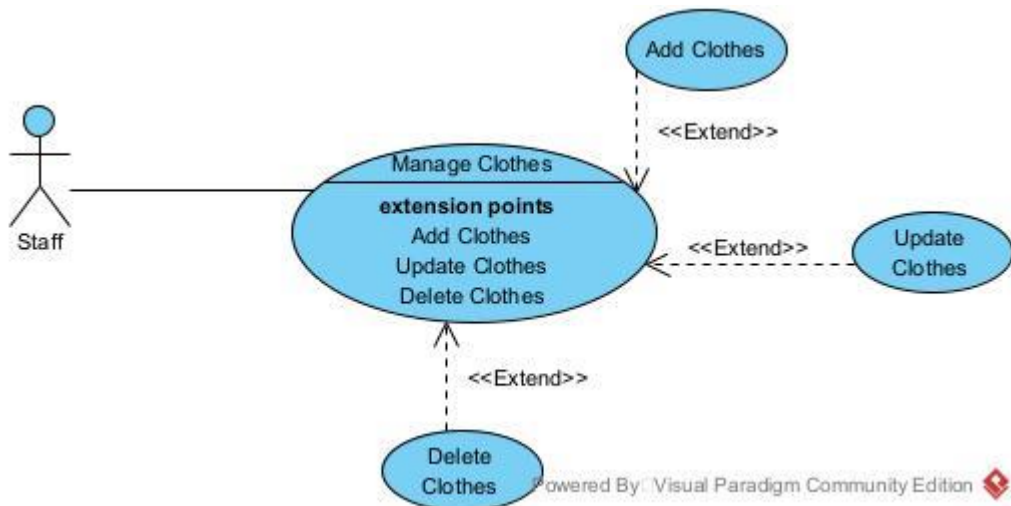
Biểu đồ usecase chi tiết order



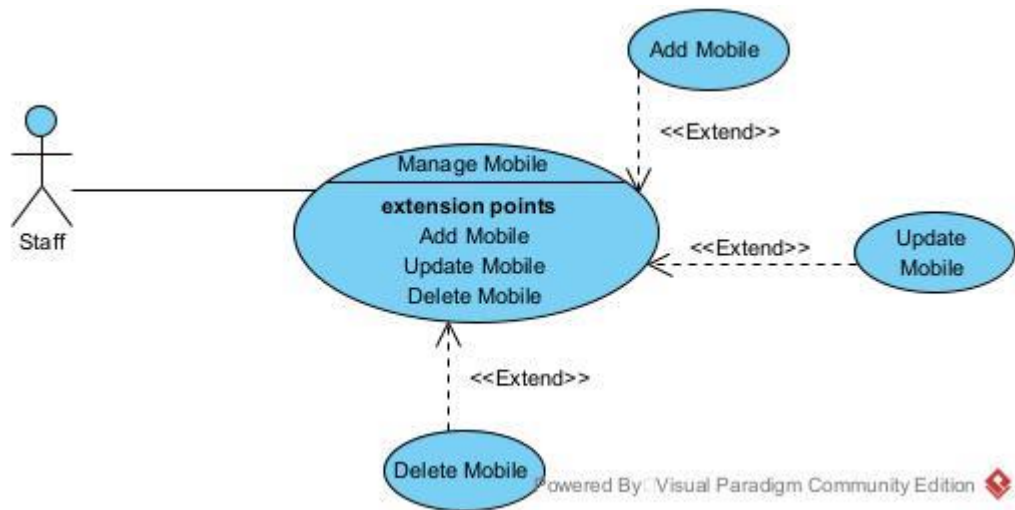
Biểu đồ usecase chi tiết manage Book



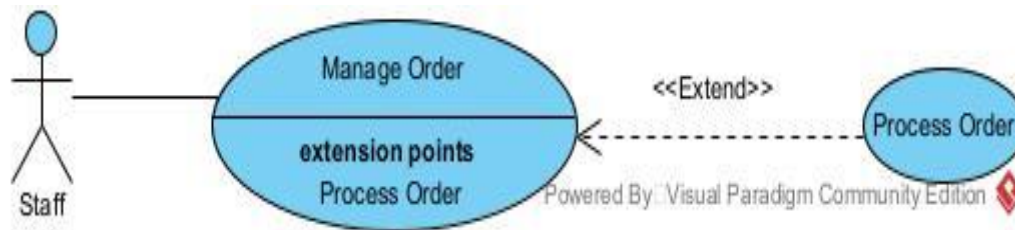
Biểu đồ usecase chi tiết manage Clothes



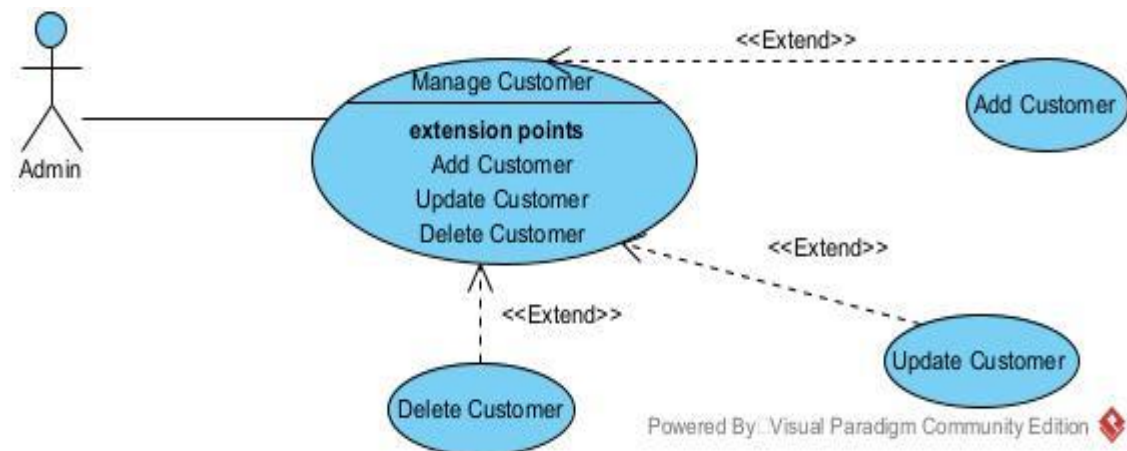
Biểu đồ usecase chi tiết manage Mobile



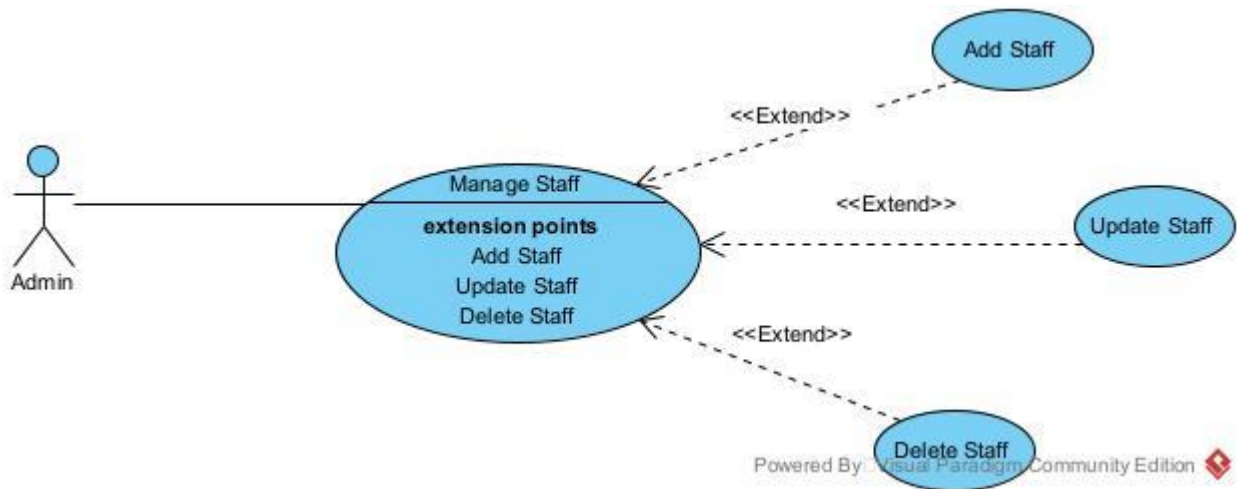
Biểu đồ usecase chi tiết manage order



Biểu đồ usecase chi tiết manage Customer



Biểu đồ usecase chi tiết manage Staff



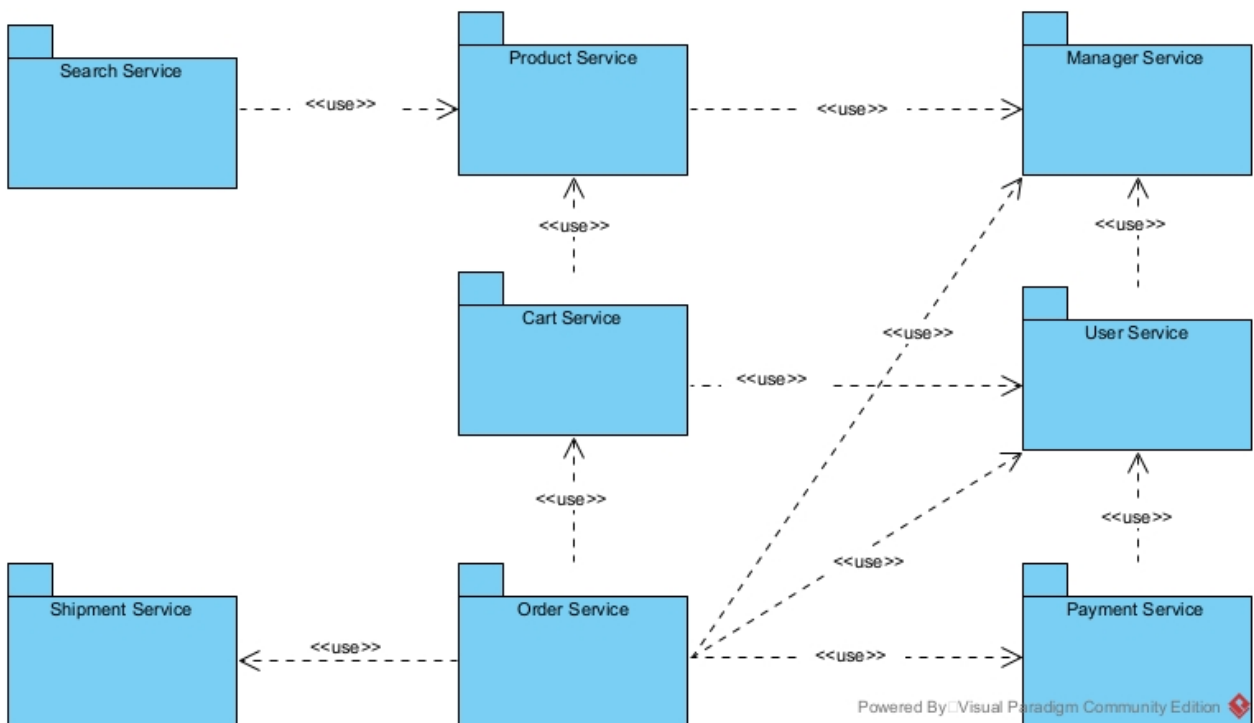
Câu 3: Vẽ biểu đồ phân rã Hệ ecomSys thành các service và các tương tác giữa các dịch vụ (sử dụng quan hệ << use >> trong UML).

❖ **Hệ ecomSys sẽ được phân rã thành các service như sau:**

- **Dịch vụ người dùng:** Dịch vụ này liên quan đến việc quản lý tài khoản người dùng và xác thực, chẳng hạn như cho phép người dùng tạo và đăng nhập vào tài khoản của họ cũng như lưu trữ thông tin hồ sơ người dùng.
- **Dịch vụ sản phẩm:** Dịch vụ này liên quan đến việc quản lý sản phẩm có sẵn để bán trên trang web, chẳng hạn như thêm sản phẩm mới, cập nhật sản phẩm, xóa sản phẩm hay xem chi tiết sản phẩm
- **Dịch vụ tìm kiếm:** Dịch vụ này liên quan đến việc tìm kiếm các sản phẩm trên trang web để xem chi tiết, thêm giỏ hàng hoặc mua sắm. Có thể tìm kiếm bằng text.
- **Dịch vụ giỏ hàng:** Dịch vụ này có khả năng chịu trách nhiệm quản lý các chức năng giỏ hàng của trang web, chẳng hạn như cho phép người dùng thêm các mặt hàng vào giỏ hàng của họ, cập nhật giỏ hàng khi các mặt hàng được thêm vào hoặc xóa và tính toán tổng chi phí của đơn hàng.
- **Dịch vụ đơn hàng:** Dịch vụ này liên quan đến quản lý quy trình đặt hàng và thực hiện đơn hàng, chẳng hạn như nhận và xử lý đơn hàng, theo dõi mức tồn kho và cập nhật cho khách hàng về trạng thái đơn hàng.
- **Dịch vụ thanh toán:** Dịch vụ này sẽ xử lý quá trình thanh toán, chẳng hạn như chấp nhận thanh toán bằng tài khoản online (ví điện tử), xác minh tính hợp lệ của thông tin thanh toán và bắt đầu giao dịch với cổng thanh toán hoặc ngân hàng.

- **Dịch vụ giao hàng:** Dịch vụ này sẽ quản lý việc vận chuyển và giao sản phẩm cho khách hàng, chẳng hạn như theo dõi các gói hàng, phối hợp với hãng vận chuyển và tính toán chi phí vận chuyển.
- **Dịch vụ nhân viên:** Dịch vụ này có thể liên quan đến việc quản lý tài khoản và quyền của nhân viên, chẳng hạn như cho phép nhân viên truy cập vào một số phần nhất định của trang web và theo dõi chỉ số hiệu suất của nhân viên.

❖ Biểu đồ phân rã



- Trong biểu đồ trên:
 - Mỗi hình chữ nhật đại diện cho một dịch vụ trong hệ thống (ví dụ: User Service, Product Service, etc.).
 - Mũi tên <<use>> biểu thị tương tác giữa các dịch vụ. Ví dụ, Cart Service có thể sử dụng Product Service để thêm sản phẩm vào giỏ hàng.
 - Các dịch vụ có thể giao tiếp với nhau theo nhiều cách, như gửi yêu cầu HTTP, sử dụng message queue, hoặc các cách khác tùy thuộc vào cấu trúc và yêu cầu của hệ thống.

Câu 4: Trình bày các dạng communication giữa các service với nhau (synchronous và asynchronous) với code và ví dụ.

❖ Các dạng communication giữa các service synchronous:

- *HTTP/HTTPS Request-Response*: Trong giao thức HTTP, dịch vụ gửi một yêu cầu (request) đến một dịch vụ khác và đợi cho đến khi nhận được một

phản hồi (response) trước khi tiếp tục thực hiện. Quá trình này là đồng bộ vì dịch vụ gửi yêu cầu phải chờ cho đến khi nhận được phản hồi.

Ví dụ:

```
import requests

def make_request():
    response = requests.get('https://api.example.com/data')
    return response.text
```

- *Remote Procedure Call (RPC)*: RPC là một phương thức giao tiếp mà một dịch vụ gửi một yêu cầu cho một dịch vụ khác để thực thi một hàm hoặc phương thức cụ thể và chờ đợi kết quả trước khi tiếp tục thực hiện. Quá trình này cũng là đồng bộ vì dịch vụ gọi trực tiếp chờ đợi cho đến khi hàm hoặc phương thức được thực thi và kết quả được trả về.

Ví dụ:

```
import xmlrpc.client

def rpc_call():
    proxy = xmlrpc.client.ServerProxy("http://localhost:8000/")
    result = proxy.add(4, 5)
    return result
```

- *WebSocket Communication*: Trong môi trường WebSocket, dịch vụ có thể thiết lập một kết nối hai chiều và gửi và nhận dữ liệu một cách liên tục. Mặc dù WebSocket cho phép giao tiếp real-time, nhưng quá trình gửi và nhận dữ liệu vẫn là đồng bộ.

Ví dụ:

```
import websocket

def websocket_communication():
    ws = websocket.create_connection("ws://localhost:8000/")
    ws.send("Hello, server!")
    result = ws.recv()
    ws.close()
    return result
```

❖ Các dạng communication giữa các service asynchronous:

- *Message Queues*: Trong hệ thống hàng đợi tin nhắn, dịch vụ gửi tin nhắn đến một hàng đợi mà không cần chờ đợi cho đến khi tin nhắn được xử lý. Dịch vụ khác có thể nhận và xử lý các tin nhắn từ hàng đợi một cách không đồng bộ, có nghĩa là chúng có thể tiếp tục thực hiện các công việc khác trong khi đang chờ đợi tin nhắn.

Ví dụ:

```
import pika

def send_message_to_queue():
    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()
    channel.queue_declare(queue='task_queue')
    channel.basic_publish(exchange='',
                          routing_key='task_queue',
                          body='Hello, queue!')
    connection.close()

def receive_message_from_queue():
    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()
    method_frame, header_frame, body = channel.basic_get(queue='task_queue')
    if method_frame:
        channel.basic_ack(method_frame.delivery_tag)
        return body.decode('utf-8')
    else:
        return None
    connection.close()
```

- *Event Streams*: Trong một môi trường dữ liệu dòng sự kiện, dịch vụ có thể gửi và nhận các sự kiện một cách không đồng bộ. Các sự kiện có thể được gửi và xử lý ngay khi chúng xảy ra mà không cần chờ đợi cho đến khi dịch vụ khác phản hồi.

Ví dụ:

```
import asyncio

async def event_stream_consumer():
    while True:
        event = await get_event_from_stream()
        process_event(event)

async def get_event_from_stream():
    # Code to fetch event from stream asynchronously
    pass

def process_event(event):
    # Code to process event
    pass
```

- *Callback Functions*: Trong một số trường hợp, các dịch vụ có thể sử dụng callback functions để thực hiện giao tiếp không đồng bộ. Dịch vụ gửi yêu cầu và chỉ định một hàm callback để xử lý kết quả khi nó được trả về.

Ví dụ:

```
import requests

def make_async_request(callback):
    response = requests.get('https://api.example.com/data')
    callback(response.text)

def process_response(response):
    # Code to process response
    pass

make_async_request(process_response)
```

Câu 5: Trình bày sử dụng các dạng communication giữa các service với code cho hệ ecomSys

❖ *Giao tiếp giữa search service vs book service, mobile service và clothes service*

```
views.py
search_service > search_engine > views.py > ...
1 from django.shortcuts import render
2 from rest_framework import viewsets
3 from rest_framework.response import Response
4 from rest_framework import status
5 from rest_framework.decorators import action
6 import requests
7
8
9 # Create your views here.
10 class SearchEngineViewSet(viewsets.ViewSet):
11
12     @action(methods=["GET"], detail=False, url_path="search/(?P<keywords>+)"
13     def search(self, request, keywords=None):
14         response = {
15             "keywords": keywords,
16             "results": [],
17         }
18         books = requests.get("http://localhost:8002/api/v1/books/").json()
19         clothes = requests.get("http://localhost:8003/api/v1/clothes/").json()
20         mobiles = requests.get("http://localhost:8004/api/v1/mobiles/").json()
21         print(books)
22         for book in books:
23             if (
24                 keywords in str(book.get("name")).lower()
25                 or keywords in str(book.get("author")).lower()
26                 or keywords in str(book.get("description")).lower()
27             ):
28                 response.get("results").append(book)
29
30         for cloth in clothes:
31             if (
32                 keywords in str(cloth.get("name")).lower()
33                 or keywords in str(cloth.get("brand")).lower()
34                 or keywords in str(cloth.get("description")).lower()
35             ):
36                 response.get("results").append(cloth)
37
```

❖ *Giao tiếp giữa cart service vs book service, clothes service và mobile service*

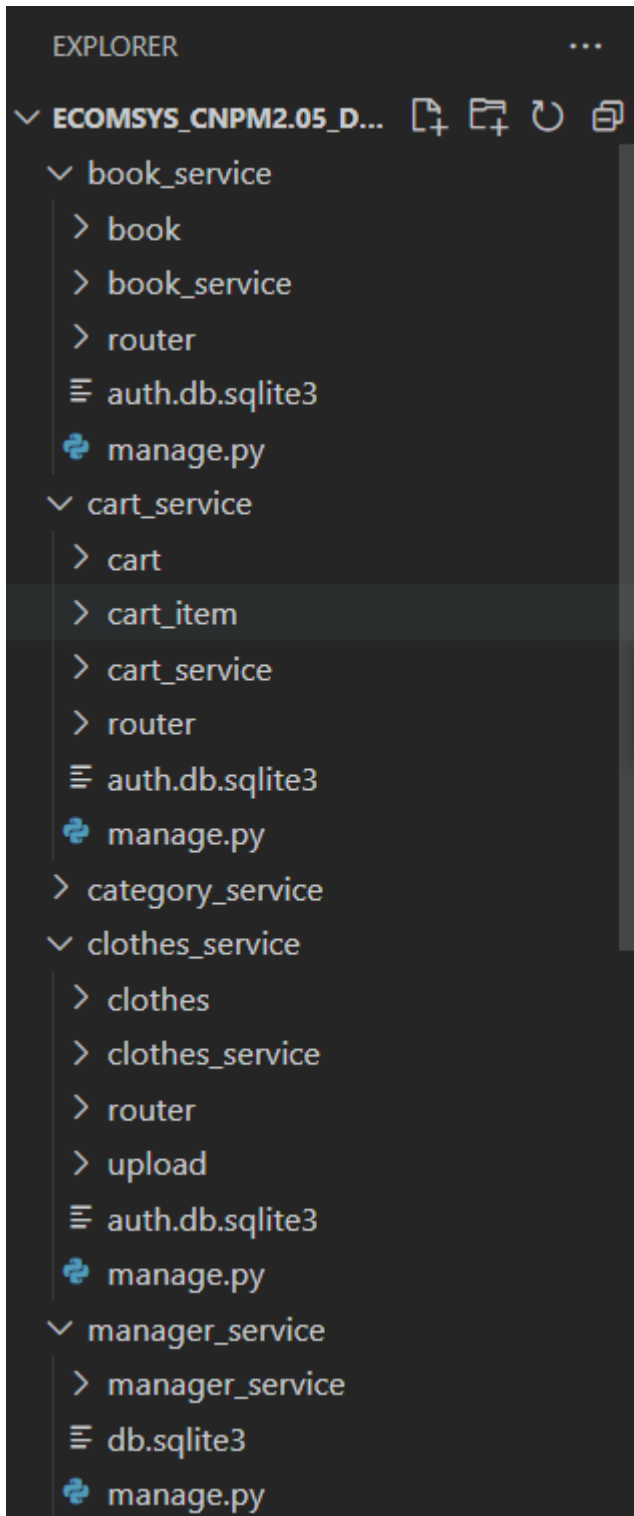
```
EXPLORER
ecomSYS_CNPM2.05.DO_DUY_KIEN
├── cart_service
│   ├── cart
│   │   ├── __pycache__
│   │   ├── migrations
│   │   ├── __init__.py
│   │   ├── admin.py
│   │   ├── apps.py
│   │   ├── models.py
│   │   ├── serializers.py
│   │   ├── tests.py
│   │   ├── urls.py
│   │   └── views.py
│   ├── cart_item
│   │   ├── __pycache__
│   │   ├── migrations
│   │   ├── __init__.py
│   │   ├── admin.py
│   │   ├── apps.py
│   │   ├── models.py
│   │   ├── serializers.py
│   │   ├── tests.py
│   │   ├── urls.py
│   │   └── views.py
│   ├── cart_service
│   │   ├── __pycache__
│   │   ├── __init__.py
│   │   ├── asgi.py
│   │   ├── settings.py
│   │   └── urls.py
│   └── OUTLINE
│       ├── TIMELINE
│       └── SERVERS
├── ...
└── ...

cart_service > cart > views.py > ...
13 # Create your views here.
14 class CartViewSet(viewsets.ViewSet):
15
16     def retrieve(self, request, pk=None):
17         cart = get_object_or_404(Cart.objects.all(), pk=pk)
18         cart_items = get_list_or_404(CartItem.objects.all(), cart=cart)
19         cart_total = Decimal("0.00")
20         for cart_item in cart_items:
21             if int(cart_item.category_id) == 1:
22                 response = requests.get(
23                     f"http://localhost:8002/api/v1/books/{cart_item.category_id}"
24                 ).json()
25                 cart_total += cart_item.quantity * Decimal(response.get("results")[0].get("price"))
26             elif int(cart_item.category_id) == 2:
27                 response = requests.get(
28                     f"http://localhost:8003/api/v1/clothes/{cart_item.category_id}"
29                 ).json()
30                 cart_total += cart_item.quantity * Decimal(response.get("results")[0].get("price"))
31             elif int(cart_item.category_id) == 3:
32                 response = requests.get(
33                     f"http://localhost:8004/api/v1/mobiles/{cart_item.category_id}"
34                 ).json()
35                 cart_total += cart_item.quantity * Decimal(response.get("results")[0].get("price"))
36
37         response = {
38             "cart": CartSerializer(cart).data,
39             "items": CartItemSerializer(cart_items, many=True).data,
40             "total": cart_total,
41         }
42         return Response(
43             data=response,
44             status=status.HTTP_200_OK,
45             content_type="application/json",
46         )
47
48 @action(methods=["GET"], detail=False, url_path="users/(?P<username>+)"
49 def users(self, request, username=None):
50
```

```
urls.py
cart_service > cart > urls.py > ...
1 from rest_framework.routers import DefaultRouter
2 from .views import CartViewSet
3
4 router = DefaultRouter()
5 router.register(r"cart", CartViewSet, basename="cart")
6
7 urlpatterns = router.urls
8
```

Project: **ecomSys_cnpm2.05_DoDuyKien**

1. 10 services in Django: *user, manager, cart, order, search, book, mobile, clothes, shipment, payment*

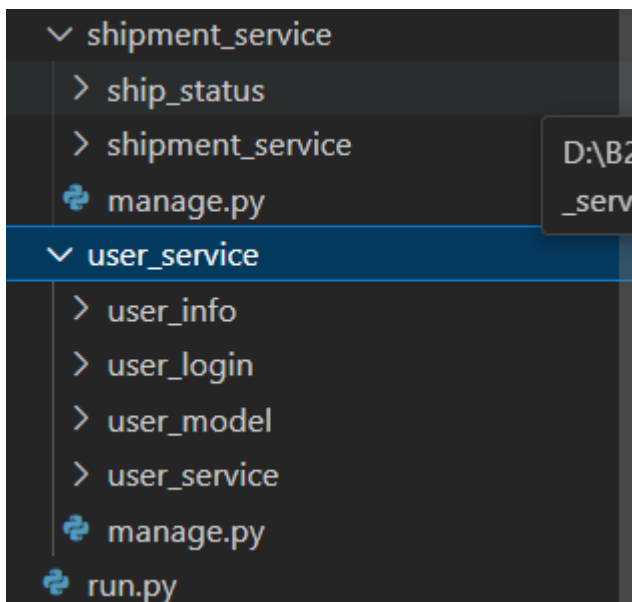


- ▼ mobile_service
 - > mobile
 - > mobile_service
 - > router
 - > upload
 - ≡ auth.db.sqlite3
 - 🔗 manage.py

- ▼ order_service
 - > order
 - > order_service
 - > router
 - ≡ db.sqlite3
 - 🔗 manage.py

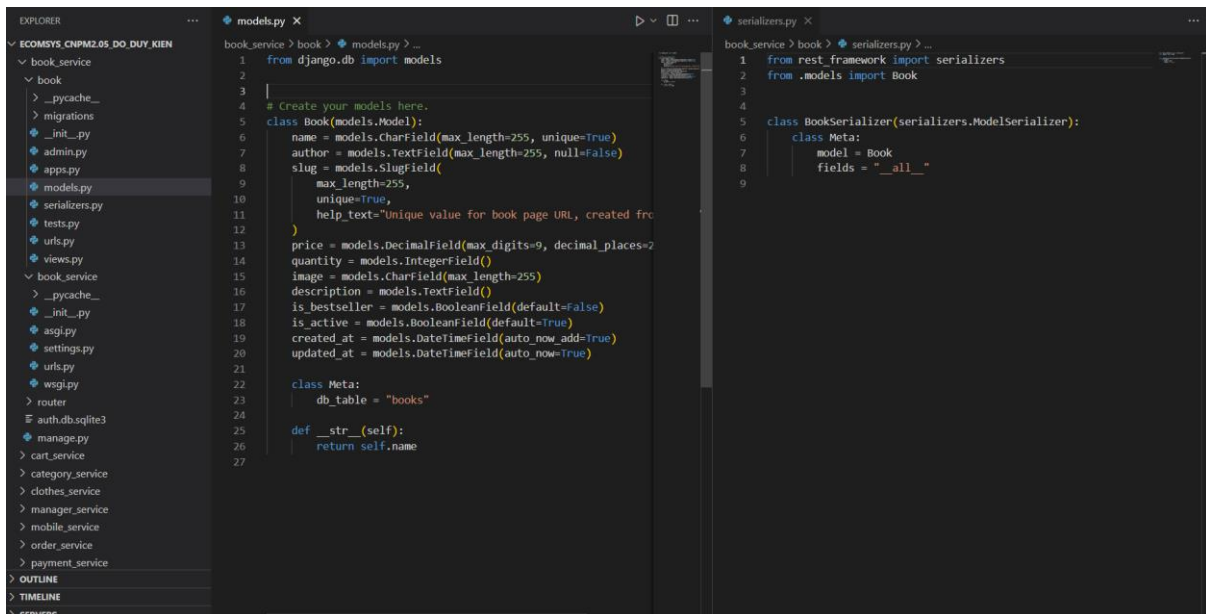
- ▼ payment_service
 - > payment
 - > payment_service
 - > router
 - > shipment_update
 - ≡ auth.db.sqlite3
 - 🔗 manage.py

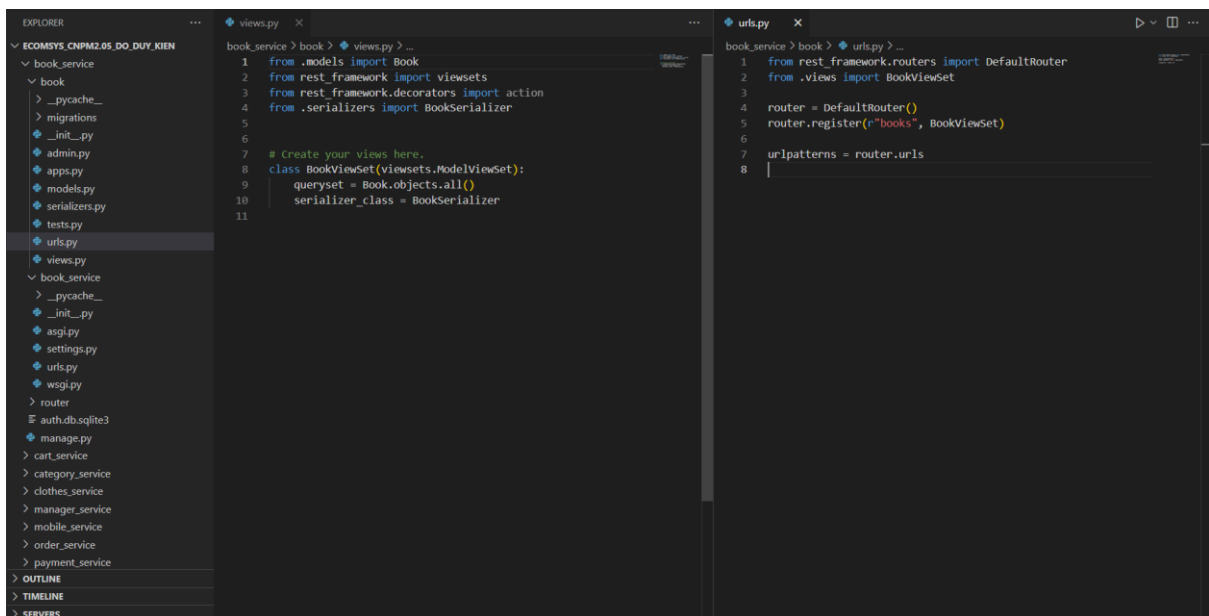
- ▼ search_service
 - > search_engine
 - > search_service
 - ≡ db.sqlite3
 - 🔗 manage.py



2. Rest API to connect to the necessary services

- book_service





- API to create book: <http://localhost:8002/api/v1/books/>

POST api/v1/books

Django / book_services / api/v1/books

POST {{BOOK_API_PREFIX}}/books/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value	Content-Type	Description	Bulk Edit
<input checked="" type="checkbox"/> name	Text Harry Potter and The Prisoner of Azkaban	Auto		
<input checked="" type="checkbox"/> author	Text J.K.Rowling	Auto		
<input checked="" type="checkbox"/> slug	Text harry-potter-and-the-prisoner-of-azkab...	Auto		
<input checked="" type="checkbox"/> price	Text 100.00	Auto		
<input checked="" type="checkbox"/> quantity	Text 140	Auto		
<input checked="" type="checkbox"/> image	Text Harry Potter 3.jpg	Auto		
<input checked="" type="checkbox"/> description	Text Interesting	Auto		

Body Cookies Headers (10) Test Results Status: 201 Created Time: 1047 ms Size: 691 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "name": "Harry Potter and The Prisoner of Azkaban",
4   "author": "J.K.Rowling",
5   "slug": "harry-potter-and-the-prisoner-of-azkaban",
6   "price": "100.00",
7   "quantity": 140,
8   "image": "Harry Potter 3.jpg",
9   "description": "Interesting",
10  "is_available": false
11 }
```

- API to get all books: <http://localhost:8002/api/v1/books/>

GET api/v1/books/ + No environment

Django / book_services / api/v1/books/ Save

GET {{BOOK_API_PREFIX}}/books/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
-----	-------	-------------	-----------

Body Cookies Headers (10) Test Results Status: 200 OK Time: 20 ms Size: 1.36 KB Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "name": "Harry Potter and The Philosophy Stone",
5     "author": "J.K.Rowling",
6     "slug": "harry-potter-and-the-philosophy-stone",
7     "price": "100.00",
8     "quantity": 123,
9     "image": "Harry Potter 1.jpg",
10    "description": "Interesting",
11    "is_bestseller": false,
12    "is_active": true,
13    "created_at": "2024-03-16T09:34:12.798000-07:00",
14    "updated_at": "2024-03-16T09:34:12.798000-07:00"
15  },
16  {
17    "id": 2,
18    "name": "Harry Potter and The Chamber Of Secret",
19    "author": "J.K.Rowling",
20    "slug": "harry-potter-and-the-chamber-of-secret",
21    "price": "100.00",
22    "quantity": 130,
23    "image": "Harry Potter 2.jpg",
```

- API to get book by ID: <http://localhost:8002/api/v1/books/{id}/>

GET api/v1/books/{id} + No environment

Django / book_services / api/v1/books/{id} Save

GET {{BOOK_API_PREFIX}}/books/1 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

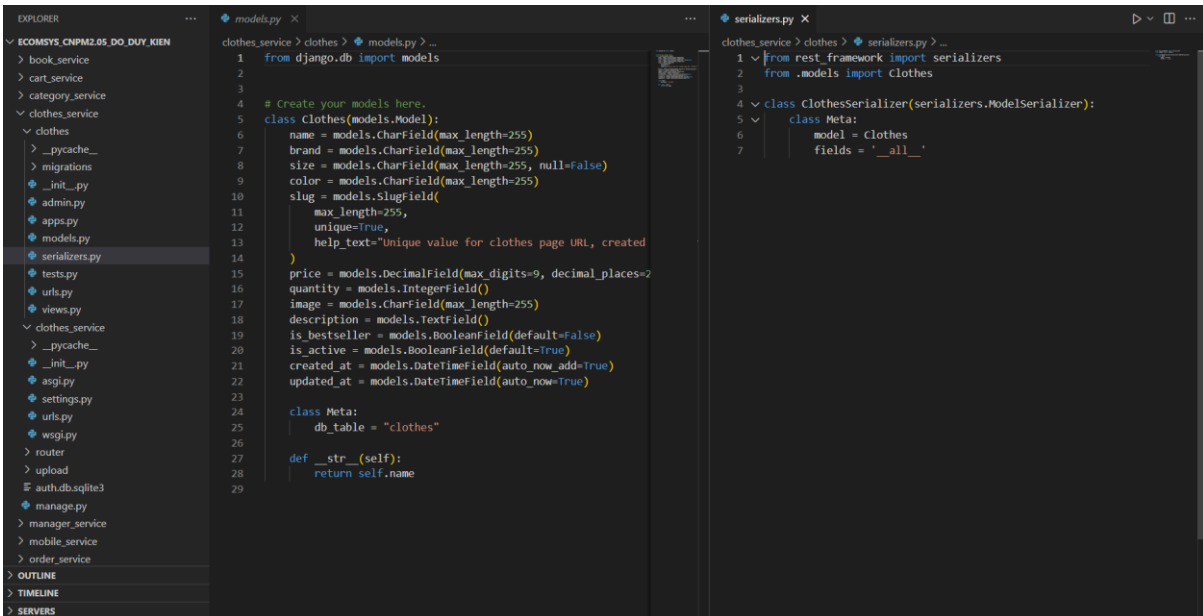
Body Cookies Headers (10) Test Results Status: 200 OK Time: 26 ms Size: 693 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "name": "Harry Potter and The Philosophy Stone",
5     "author": "J.K.Rowling",
6     "slug": "harry-potter-and-the-philosophy-stone",
7     "price": "100.00",
8     "quantity": 123,
9     "image": "Harry Potter 1.jpg",
10    "description": "Interesting",
11    "is_bestseller": false,
12    "is_active": true,
13    "created_at": "2024-03-16T09:34:12.798000-07:00",
14    "updated_at": "2024-03-16T09:34:12.798000-07:00"
15  }
```

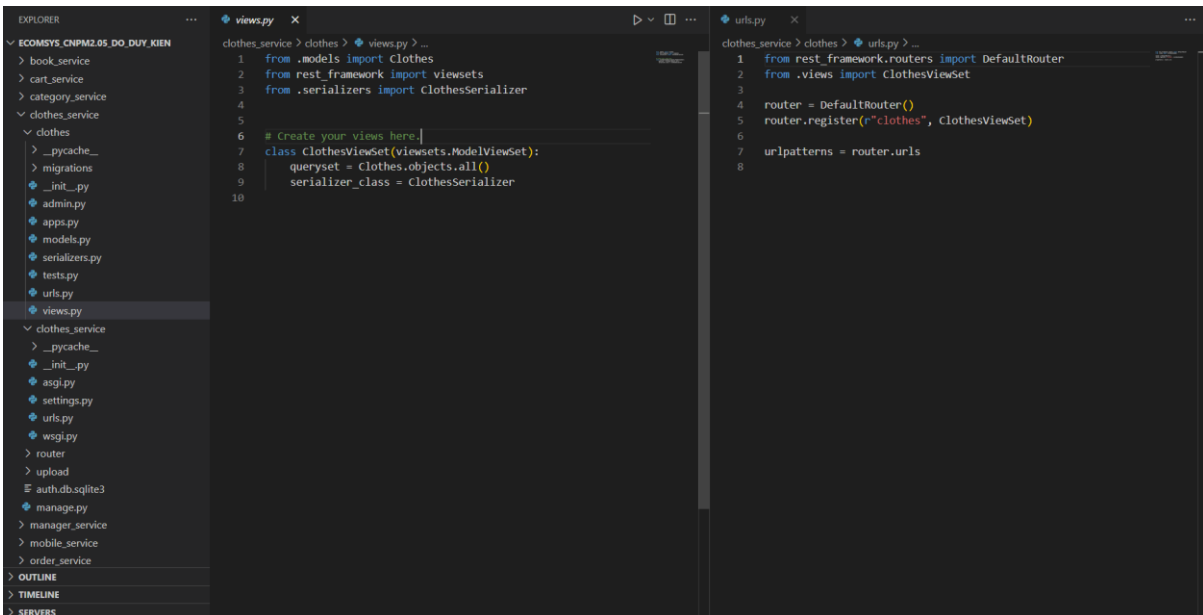
- clothes_service

```
78 DATABASES = {
79     'default': {},
80     'auth_db': {
81         'ENGINE': 'django.db.backends.sqlite3',
82         'NAME': BASE_DIR / 'auth.db.sqlite3',
83     },
84     'mongodb': {
85         'ENGINE': 'djongo',
86         'NAME': 'clothes_service',
87     },
88 }
89
90 DATABASE_ROUTERS = ['router.database_routers.AuthDBRouter', 'router.database_routers.MongoDBRouter']
91
```



The screenshot shows the Visual Studio Code editor with two files open: `models.py` and `serializers.py`. The `models.py` file defines a `Clothes` model with fields for name, brand, size, color, slug, price, quantity, image, description, is_bestseller, is_active, created_at, and updated_at. The `serializers.py` file defines a `ClothesSerializer` class that inherits from `ModelSerializer` and uses the `Clothes` model.

```
models.py
1 from django.db import models
2
3
4 # Create your models here.
5 class Clothes(models.Model):
6     name = models.CharField(max_length=255)
7     brand = models.CharField(max_length=255)
8     size = models.CharField(max_length=255, null=False)
9     color = models.CharField(max_length=255)
10    slug = models.SlugField(
11        max_length=255,
12        unique=True,
13        help_text="Unique value for clothes page URL, created
14    )
15    price = models.DecimalField(max_digits=9, decimal_places=2)
16    quantity = models.IntegerField()
17    image = models.CharField(max_length=255)
18    description = models.TextField()
19    is_bestseller = models.BooleanField(default=False)
20    is_active = models.BooleanField(default=True)
21    created_at = models.DateTimeField(auto_now_add=True)
22    updated_at = models.DateTimeField(auto_now=True)
23
24    class Meta:
25        db_table = "clothes"
26
27    def __str__(self):
28        return self.name
29
serializers.py
1 from rest_framework import serializers
2 from .models import Clothes
3
4 class ClothesSerializer(serializers.ModelSerializer):
5     class Meta:
6         model = Clothes
7         fields = '__all__'
```



The screenshot shows the Visual Studio Code editor with two files open: `views.py` and `urls.py`. The `views.py` file defines a `ClothesViewSet` class that inherits from `ModelViewSet` and uses the `Clothes` model and `ClothesSerializer`. The `urls.py` file defines a `DefaultRouter` and registers the `ClothesViewSet` under the `clothes` namespace.

```
views.py
1 from .models import Clothes
2 from rest_framework import viewsets
3 from .serializers import ClothesSerializer
4
5 # Create your views here.
6 class ClothesViewSet(viewsets.ModelViewSet):
7     queryset = Clothes.objects.all()
8     serializer_class = ClothesSerializer
9
urls.py
1 from rest_framework.routers import DefaultRouter
2 from .views import ClothesViewSet
3
4 router = DefaultRouter()
5 router.register(r"clothes", ClothesViewSet)
6
7 urlpatterns = router.urls
```

- API to create clothes: <http://localhost:8003/api/v1/clothes/>

POST /api/v1/clothes/ + No environment

Django / clothes_services / /api/v1/clothes/ Save

POST {{CLOTHES_API_PREFIX}}/clothes/ Send

Params Authorization Request Script Tests Settings Cookies

☐ none ☒ form-data

INITIAL http://127.0.0.1:8003/api/v1
CURRENT http://127.0.0.1:8003/api/v1
SCOPE Global

Key	Value	Content-Type	Description	Bulk Edit
<input checked="" type="checkbox"/> name	áo khoác nữ	Auto		
<input checked="" type="checkbox"/> brand	Text TokyoLife	Auto		
<input checked="" type="checkbox"/> size	Text M	Auto		
<input checked="" type="checkbox"/> color	Text Blue	Auto		
<input checked="" type="checkbox"/> slug	Text ao-khoac-nu	Auto		
<input checked="" type="checkbox"/> price	Text 50	Auto		
<input checked="" type="checkbox"/> quantity	Text 100	Auto		
<input checked="" type="checkbox"/> image	Text Woman Jacket 1.jpg	Auto		

Body Cookies Headers (10) Test Results Status: 201 Created Time: 344 ms Size: 652 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 2,
3   "name": "Áo khoác nữ",
4   "brand": "TokyoLife",
5   "size": "M",
6   "color": "Blue",
7   "slug": "ao-khoac-nu",
8   "price": "50.00".
}
```

Postbot Runner Start Proxy Cookies Trash

- API to get all clothes: <http://localhost:8003/api/v1/clothes/>

GET /api/v1/clothes/ + No environment

Django / clothes_services / /api/v1/clothes/ Save

GET {{CLOTHES_API_PREFIX}}/clothes/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
-----	-------	-------------	-----------

Body Cookies Headers (10) Test Results Status: 200 OK Time: 27 ms Size: 967 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "name": "Áo khoác nam",
5     "brand": "TokyoLife",
6     "size": "XL",
7     "color": "Black",
8     "slug": "ao-khoac-nam",
9     "price": "50.00",
10    "quantity": 100,
11    "image": "Man Jacket 1.jpg",
12    "description": "Pretty",
13    "is_bestseller": true,
14    "is_active": true,
15    "created_at": "2024-03-16T21:58:07.960000-07:00",
16    "updated_at": "2024-03-16T21:58:07.960000-07:00"
17  },
18  {
19    "id": 2,
20    "name": "Áo khoác nữ",
21    "brand": "TokyoLife",
22    "size": "M",
23    "color": "Blue",

```

Postbot Runner Start Proxy Cookies Trash

- API to get clothes by ID: <http://localhost:8003/api/v1/clothes/{id}/>

GET /api/v1/clothes/{id} + No environment

Django / clothes_services / /api/v1/clothes/{id} Save

GET {{CLOTHES_API_PREFIX}}/clothes/1 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 27 ms Size: 661 B Save as example

Pretty Raw Preview Visualize JSON

```

1  {
2    "id": 1,
3    "name": "Áo khoác nam",
4    "brand": "TokyoLife",
5    "size": "XL",
6    "color": "Black",
7    "slug": "ao-khoac-nam",
8    "price": "50.00",
9    "quantity": 100,
10   "image": "Man Jacket 1.jpg",
11   "description": "Pretty",
12   "is_bestseller": true,
13   "is_active": true,
14   "created_at": "2024-03-16T21:58:07.960000-07:00",
15   "updated_at": "2024-03-16T21:58:07.960000-07:00"
16 }

```

Postbot Runner Start Proxy Cookies Trash

- mobile_service

```

78 DATABASES = {
79     'default': {},
80     'auth_db': {
81         'ENGINE': 'django.db.backends.sqlite3',
82         'NAME': BASE_DIR / 'auth.db.sqlite3',
83     },
84     'mongodb': {
85         'ENGINE': 'djongo',
86         'NAME': 'mobile_service',
87     },
88 }
89
90 DATABASE_ROUTERS = ['router.database_routers.AuthDBRouter', 'router.database_routers.MongoDBRouter']
91

```

The screenshot shows two files in a Django project: `models.py` and `serializers.py`. The `models.py` file defines a `Mobile` model with fields for name, brand, slug, price, quantity, image, description, and active status. The `serializers.py` file defines a `MobileSerializer` class that inherits from `ModelSerializer` and uses the `Mobile` model.

```
models.py
1 from django.db import models
2
3
4 # Create your models here.
5 class Mobile(models.Model):
6     name = models.CharField(max_length=255)
7     brand = models.CharField(max_length=255, null=False)
8     slug = models.SlugField(
9         max_length=255,
10        unique=True,
11        help_text="Unique value for mobile page URL, created f
12    )
13     price = models.DecimalField(max_digits=9, decimal_places=2)
14     quantity = models.IntegerField()
15     image = models.CharField(max_length=255)
16     description = models.TextField()
17     is_bestseller = models.BooleanField(default=False)
18     is_active = models.BooleanField(default=True)
19     created_at = models.DateTimeField(auto_now_add=True)
20     updated_at = models.DateTimeField(auto_now=True)
21
22
23 class Meta:
24     db_table = "mobiles"
25
26 def __str__(self):
27     return self.name
28
29
30 serializers.py
31 from rest_framework import serializers
32 from .models import Mobile
33
34
35 class MobileSerializer(serializers.ModelSerializer):
36
37     class Meta:
38         model = Mobile
39         fields = "__all__"
```

The screenshot shows two files in a Django project: `views.py` and `urls.py`. The `views.py` file defines a `MobileViewSet` class that inherits from `ModelViewSet` and uses the `Mobile` model and `MobileSerializer`. The `urls.py` file defines the `urlpatterns` for the mobile application, including the `MobileViewSet`.

```
views.py
1 from .models import Mobile
2 from rest_framework import viewsets
3
4
5 # Create your views here.
6 class MobileViewSet(viewsets.ModelViewSet):
7     queryset = Mobile.objects.all()
8     serializer_class = MobileSerializer
9
10
11 urls.py
12 from rest_framework.routers import DefaultRouter
13 from .views import MobileViewSet
14
15
16 router = DefaultRouter()
17 router.register(r'mobiles', MobileViewSet)
18
19 urlpatterns = router.urls
```

- API to create mobile: <http://localhost:8004/api/v1/mobiles/>

POST /api/v1/mobiles/ + No environment

Django / mobile_services / /api/v1/mobiles/ Save

POST {{MOBILE_API_PREFIX}}/mobiles/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	Key		Value	Content-Type	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	name	Text	Iphone 15 Promax	Auto			
<input checked="" type="checkbox"/>	brand	Text	Apple	Auto			
<input checked="" type="checkbox"/>	slug	Text	iphone-15-promax	Auto			
<input checked="" type="checkbox"/>	price	Text	2000	Auto			
<input checked="" type="checkbox"/>	quantity	Text	20	Auto			
<input checked="" type="checkbox"/>	image	Text	Iphone 15.jpg	Auto			
<input checked="" type="checkbox"/>	description	Text	Brand new product from Apple	Auto			
<input checked="" type="checkbox"/>	is_bestseller	Text	True	Auto			
<input checked="" type="checkbox"/>	is_active	Text	True	Auto			

Body Cookies Headers (10) Test Results Status: 201 Created Time: 375 ms Size: 646 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 2,
3   "name": "Iphone 15 Promax",
4   "brand": "Apple",
5   "slug": "iphone-15-promax",
6   "price": "2000.00",
```

- API to get all mobiles: <http://localhost:8004/api/v1/mobiles/>

GET /api/v1/mobiles/ + No environment

Django / mobile_services / /api/v1/mobiles/ Save

GET {{MOBILE_API_PREFIX}}/mobiles/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body Cookies Headers (10) Test Results Status: 200 OK Time: 24 ms Size: 955 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "name": "Iphone 14 Promax",
5     "brand": "Apple",
6     "slug": "iphone-14-promax",
7     "price": "1000.00",
8     "quantity": 50,
9     "image": "Iphone 14.jpg",
10    "description": "Brand new product from Apple",
11    "is_bestseller": true,
12    "is_active": true,
13    "created_at": "2024-03-16T22:02:07.420000-07:00",
14    "updated_at": "2024-03-16T22:02:07.420000-07:00"
15  },
16  {
17    "id": 2,
18    "name": "Iphone 15 Promax",
19    "brand": "Apple",
20    "slug": "iphone-15-promax",
21    "price": "2000.00",
22    "quantity": 20,
23    "image": "Iphone 15.jpg",
```


- API to get mobile by ID: <http://localhost:8004/api/v1/mobiles/{id}/>

GET /api/v1/mobiles/{id} + No environment

Django / mobile_services / /api/v1/mobiles/{id}

GET {{(MOBILE_API_PREFIX)}/mobiles/1 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (10) Test Results Status: 200 OK Time: 21 ms Size: 655 B Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 1,
3   "name": "Iphone 14 Promax",
4   "brand": "Apple",
5   "slug": "iphone-14-promax",
6   "price": "1000.00",
7   "quantity": 50,
8   "image": "Iphone 14.jpg",
9   "description": "Brand new product from Apple",
10  "is_bestseller": true,
11  "is_active": true,
12  "created_at": "2024-03-16T22:02:07.420000-07:00",
13  "updated_at": "2024-03-16T22:02:07.420000-07:00"
14 }
```

Postbot Runner Start Proxy Cookies Trash

- search_service

EXPLORER views.py

```

9 # Create your views here.
10 class SearchEngineViewSet(viewsets.ViewSet):
11
12     @action(methods=["GET"], detail=False, url_path="search/{keywords}")
13     def search(self, request, keywords=None):
14         response = {
15             "keywords": keywords,
16             "results": [],
17         }
18
19         books = requests.get("http://localhost:8002/api/v1/books").json()
20         clothes = requests.get("http://localhost:8003/api/v1/clothes").json()
21         mobiles = requests.get("http://localhost:8004/api/v1/mobiles").json()
22         print(books)
23         for book in books:
24             if (
25                 keywords in str(book.get("name")).lower()
26                 or keywords in str(book.get("author")).lower()
27                 or keywords in str(book.get("description")).lower()
28             ):
29                 response.get("results").append(book)
30
31         for cloth in clothes:
32             if (
33                 keywords in str(cloth.get("name")).lower()
34                 or keywords in str(cloth.get("brand")).lower()
35                 or keywords in str(cloth.get("description")).lower()
36             ):
37                 response.get("results").append(cloth)
38
39         for mobile in mobiles:
40             if (
41                 keywords in str(mobile.get("name")).lower()
42                 or keywords in str(mobile.get("brand")).lower()
43                 or keywords in str(mobile.get("description")).lower()
44             ):
45                 response.get("results").append(mobile)
46
47         return response
48 
```

urls.py

```

1 from rest_framework.routers import DefaultRouter
2 from .views import SearchEngineViewSet
3
4 router = DefaultRouter()
5 router.register(r'', SearchEngineViewSet, basename="search-engine")
6
7 urlpatterns = router.urls
8 
```

- API to search products: <http://localhost:8009/api/v1/search/{keywords}>

GET api/v1/search/{keywords} + No environment

Django / search_services / api/v1/search/{keywords}

GET {{SEARCH_API_PREFIX}}/search/iphone Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body Cookies Headers (10) Test Results Status: 200 OK Time: 6.16 s Size: 981 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "keywords": "iphone",
3   "results": [
4     {
5       "id": 1,
6       "name": "Iphone 14 Promax",
7       "brand": "Apple",
8       "slug": "iphone-14-promax",
9       "price": "1000.00",
10      "quantity": 50,
11      "image": "Iphone 14.jpg",
12      "description": "Brand new product from Apple",
13      "is_bestseller": true,
14      "is_active": true,
15      "created_at": "2024-03-16T22:02:07.420000-07:00",
16      "updated_at": "2024-03-16T22:02:07.420000-07:00"
17    },
18    {
19      "id": 2,
20      "name": "Iphone 15 Promax",
21      "brand": "Apple",
22      "slug": "iphone-15-promax",
23      "price": "2000.00",
```

Postbot Runner Start Proxy Cookies Trash

- cart_service
- cart

EXPLORER

models.py

```
1 from django.db import models
2
3
4 # Create your models here.
5 class Cart(models.Model):
6     user_id = models.CharField(max_length=255)
7     status = models.BooleanField(default=False, auto_created=True)
8     created_at = models.DateTimeField(auto_now_add=True)
9     updated_at = models.DateTimeField(auto_now=True)
10
11     class Meta:
12         db_table = "carts"
13
```

serializers.py

```
1 from rest_framework import serializers
2 from .models import Cart
3
4
5 class CartSerializer(serializers.ModelSerializer):
6     class Meta:
7         model = Cart
8         fields = "__all__"
9
```

The screenshot shows two editor windows in VS Code. The left window, titled 'views.py', contains the following code:

```
13 # Create your views here.
14 class CartViewSet(viewsets.ViewSet):
15
16     def retrieve(self, request, pk=None):
17         cart = get_object_or_404(Cart.objects.all(), pk=pk)
18         cart_items = get_list_or_404(Cartitem.objects.all(), cart=cart)
19         cart_total = Decimal("0.00")
20         for cart_item in cart_items:
21             if int(cart_item.category_id) == 1:
22                 response = requests.get(
23                     f"http://localhost:8002/api/v1/books/{cart_item.product_id}")
24                 .json()
25                 cart_total += cart_item.quantity * Decimal(response['price'])
26             elif int(cart_item.category_id) == 2:
27                 response = requests.get(
28                     f"http://localhost:8003/api/v1/clothes/{cart_item.product_id}")
29                 .json()
30                 cart_total += cart_item.quantity * Decimal(response['price'])
31             elif int(cart_item.category_id) == 3:
32                 response = requests.get(
33                     f"http://localhost:8004/api/v1/mobiles/{cart_item.product_id}")
34                 .json()
35                 cart_total += cart_item.quantity * Decimal(response['price'])
36
37         response = {
38             "cart": CartSerializer(cart).data,
39             "items": CartitemSerializer(cart_items, many=True).data,
40             "total": cart_total,
41         }
42         return Response(
43             data=response,
44             status=status.HTTP_200_OK,
45             content_type="application/json",
46         )
47
48 @action(methods=["GET"], detail=False, url_path="users/{?product_id}")
```

The right window, titled 'urls.py', contains the following code:

```
1 from rest_framework.routers import DefaultRouter
2 from .views import CartViewSet
3
4 router = DefaultRouter()
5 router.register(r"cards", CartViewSet, basename="cart")
6
7 urlpatterns = router.urls
```

- cart_item

The screenshot shows two editor windows in VS Code. The left window, titled 'models.py', contains the following code:

```
1 from django.db import models
2 from cart.models import Cart
3
4 # Create your models here.
5 class Cartitem(models.Model):
6     cart = models.ForeignKey(Cart, on_delete=models.CASCADE)
7     date_added = models.DateTimeField(auto_now_add=True)
8     category_id = models.CharField(max_length=50, unique=False)
9     product_id = models.CharField(max_length=50, unique=False, blank=True)
10     quantity = models.IntegerField(default=1, unique=False, blank=True)
11     def augment_quantity(self, quantity):
12         self.quantity = self.quantity + int(quantity)
13         self.save()
14
15 class CartitemSerializer(serializers.ModelSerializer):
16     class Meta:
17         model = Cartitem
18         fields = ('cart', 'category_id', 'product_id', 'quantity')
```

The right window, titled 'serializers.py', contains the following code:

```
1 from rest_framework import serializers
2 from .models import Cartitem
3
4 class CartitemSerializer(serializers.ModelSerializer):
5     class Meta:
6         model = Cartitem
7         fields = ('cart', 'category_id', 'product_id', 'quantity')
```

```
EXPLORER
├── ECOMSYS_CNP205_DO_DUY_KIEN
│   ├── cart_service
│   │   ├── cart
│   │   │   ├── __pycache__
│   │   │   ├── migrations
│   │   │   ├── __init__.py
│   │   │   ├── admin.py
│   │   │   ├── apps.py
│   │   │   ├── models.py
│   │   │   ├── serializers.py
│   │   │   ├── tests.py
│   │   │   ├── urls.py
│   │   │   └── views.py
│   │   ├── cart_item
│   │   │   ├── __pycache__
│   │   │   ├── migrations
│   │   │   ├── __init__.py
│   │   │   ├── admin.py
│   │   │   ├── apps.py
│   │   │   ├── models.py
│   │   │   ├── serializers.py
│   │   │   ├── tests.py
│   │   │   ├── urls.py
│   │   │   └── views.py
│   │   ├── cart_service
│   │   │   ├── __pycache__
│   │   │   ├── __init__.py
│   │   │   ├── asgi.py
│   │   │   ├── settings.py
│   │   │   └── urls.py
│   │   └── OUTLINE
│   │       ├── TIMELINE
│   │       └── SERVERS
│   └── ...
└── ...

views.py
8 # Create your views here.
9 class CartItemViewSet(viewsets.ViewSet):
10
11     def create(self, request):
12         post_data = request.data
13         serializer = CartItemSerializer(data=post_data)
14         if serializer.is_valid():
15             cart_id = post_data.get("cart")
16             category_id = post_data.get("category_id")
17             product_id = post_data.get("product_id")
18             quantity = post_data.get("quantity")
19             cart_items = CartItem.objects.filter(cart_id=cart_id)
20             book_in_cart = False
21             for cart_item in cart_items:
22                 if (
23                     cart_item.category_id == category_id
24                     and cart_item.product_id == product_id
25                 ):
26                     cart_item.increment_quantity(quantity=quantity)
27                     book_in_cart = True
28                     break
29             if not book_in_cart:
30                 serializer.save()
31
32             return Response(
33                 data={"message": "Add item to cart successful"},
34                 status=status.HTTP_200_OK,
35                 content_type="application/json",
36             )
37         else:
38             return Response(serializer.errors, status=400)
39
40     def update(self, request, pk=None):
41         update_data = request.data
42         cart_item = CartItem.objects.filter(pk=pk).get()
43         serializer = CartItemSerializer(cart_item, data=update_data)
```

- API to create cart: <http://localhost:8005/api/v1/carts/>

POST api/v1/carts/ + No environment

Django / cart_services / api/v1/carts/ Save

POST ({{CART_API_PREFIX}})/carts/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value	Content-Type	Description	Bulk Edit
<input checked="" type="checkbox"/> user_id	Text 1	Auto		
Key	Text Value	Auto	Description	

Body Cookies Headers (10) Test Results Status: 201 Created Time: 163 ms Size: 465 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "status": false,
4   "user_id": "1",
5   "created_at": "2024-03-18T05:47:32.246344-07:00",
6   "updated_at": "2024-03-18T05:47:32.246344-07:00"
7 }
```

Postbot Runner Start Proxy Cookies Trash

- API to get cart by ID: http://localhost:8005/api/v1/carts/{cart_id}

GET /api/v1/carts/{cart_id} + No environment

Django / cart_services / /api/v1/carts/{cart_id}

GET {{CART_API_PREFIX}}/carts/1/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value	Content-Type	Description	Bulk Edit
-----	-------	--------------	-------------	-----------

Body Cookies Headers (10) Test Results Status: 200 OK Time: 6.23 s Size: 679 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "cart": {
3     "id": 1,
4     "status": true,
5     "user_id": "1",
6     "created_at": "2024-03-17T04:11:39.985414-07:00",
7     "updated_at": "2024-03-17T21:02:31.922052-07:00"
8   },
9   "items": [
10    {
11      "cart": 1,
12      "category_id": "1",
13      "product_id": "1",
14      "quantity": 1
15    },
16    {
17      "cart": 1,
18      "category_id": "1",
19      "product_id": "2",
20      "quantity": 1
21    },
22    {
23      "cart": 1,
```

- API to get cart with user ID:
http://localhost:8005/api/v1/carts/users/{user_id}

GET /api/v1/carts/users/{user_id} No environment

Django / cart_services / /api/v1/carts/users/{user_id}/ Save

GET {{(CART_API_PREFIX)}}/carts/users/1 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
-----	-------	-------------	-----------

Body Cookies Headers (10) Test Results Status: 200 OK Time: 10.45 s Size: 961 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "cart": {
4       "id": 1,
5       "status": true,
6       "user_id": "1",
7       "created_at": "2024-03-17T04:11:39.985414-07:00",
8       "updated_at": "2024-03-17T21:02:31.922052-07:00"
9     },
10    "items": [
11      {
12        "cart": 1,
13        "category_id": "1",
14        "product_id": "1",
15        "quantity": 1
16      },
17      {
18        "cart": 1,
19        "category_id": "1",
20        "product_id": "2",
21        "quantity": 1
22      }
23    ]
24  }
25 ]
```

Postbot Runner Start Proxy Cookies Trash

- API to add item to cart: http://localhost:8005/api/v1/cart_items/

POST /api/v1/cart_items/ No environment

Django / cart_services / /api/v1/cart_items/ Save

POST {{(CART_API_PREFIX)}}/cart_items/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value	Content-Type	Description	Bulk Edit
<input checked="" type="checkbox"/> cart	Text 3	Auto		
<input checked="" type="checkbox"/> category_id	Text 2	Auto		
<input checked="" type="checkbox"/> product_id	Text 1	Auto		
<input checked="" type="checkbox"/> quantity	Text 2	Auto		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 19 ms Size: 370 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Add item to cart successfully."
3 }
```

Postbot Runner Start Proxy Cookies Trash

- API to update item in cart:

http://localhost:8005/api/v1/cart_items/{cart_item_id}/

The screenshot shows a REST client interface with the following details:

- URL:** `PUT /api/v1/cart_items/{cart_item_id}/`
- Environment:** No environment
- Path:** Django / cart_services / /api/v1/cart_items/{cart_item_id}
- Method:** PUT
- Body:** `{{CART_API_PREFIX}}/cart_items/1/`
- Params:** Authorization, Headers (8), Body, Pre-request Script, Tests, Settings
- Body Type:** form-data (selected), none, x-www-form-urlencoded, raw, binary, GraphQL
- Form Data:**

Key	Value
cart	1
category_id	1
product_id	1
quantity	2
- Status:** 200 OK, Time: 25 ms, Size: 372 B
- Response Body (JSON):**

```
{  "message": "Update item in cart successfully."}
```

- API to delete item in cart:

http://localhost:8005/api/v1/cart_items/{cart_item_id}

DEL api/v1/cart_items/(cart_i + No environment

Django / cart_services / api/v1/cart_items/(cart_item_id)

DELETE {{(CART_API_PREFIX)}/cart_items/6/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value	Content-Type	Description	Bulk Edit
Key	Text	Value	Auto	Description

Body Cookies Headers (10) Test Results Status: 200 OK Time: 16 ms Size: 372 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Delete item in cart successfully."
3 }
```

Postbot Runner Start Proxy Cookies Trash

3. Search process and Create cart, Add to cart

- Search process

```
views.py
search_service > search_engine > views.py > ...
1 from django.shortcuts import render
2 from rest_framework import viewsets
3 from rest_framework.response import Response
4 from rest_framework import status
5 from rest_framework.decorators import action
6 import requests
7
8
9 # Create your views here.
10 class SearchEngineViewSet(viewsets.ViewSet):
11
12     @action(methods=["GET"], detail=False, url_path="search/(?P<keywords>+.+)")
13     def search(self, request, keywords=None):
14         response = {
15             "keywords": keywords,
16             "results": [],
17         }
18         books = requests.get("http://localhost:8002/api/v1/books/").json()
19         clothes = requests.get("http://localhost:8003/api/v1/clothes/").json()
20         mobiles = requests.get("http://localhost:8004/api/v1/mobiles/").json()
21         print(books)
22         for book in books:
23             if (
24                 keywords in str(book.get("name")).lower()
25                 or keywords in str(book.get("author")).lower()
26                 or keywords in str(book.get("description")).lower()
27             ):
28                 response.get("results").append(book)
29
30         for cloth in clothes:
31             if (
32                 keywords in str(cloth.get("name")).lower()
33                 or keywords in str(cloth.get("brand")).lower()
34                 or keywords in str(cloth.get("description")).lower()
35             ):
36                 response.get("results").append(cloth)
37
```

```

38         for mobile in mobiles:
39             if (
40                 keywords in str(mobile.get("name")).lower()
41                 or keywords in str(mobile.get("brand")).lower()
42                 or keywords in str(mobile.get("description")).lower()
43             ):
44                 response.get("results").append(mobile)
45
46         return Response(
47             data=response, status=status.HTTP_200_OK, content_type="application/json"
48         )
49

```

- Create cart

```

def create(self, request):
    cart = CartSerializer(data=request.data)
    if cart.is_valid():
        cart.save()
        return Response(cart.data, status=201)
    else:
        return Response(cart.errors, status=400)

```

- Add to cart

```

def create(self, request):
    post_data = request.data
    serializer = CartItemSerializer(data=post_data)
    if serializer.is_valid():
        cart_id = post_data.get("cart")
        category_id = post_data.get("category_id")
        product_id = post_data.get("product_id")
        quantity = post_data.get("quantity")
        cart_items = CartItem.objects.filter(cart_id=cart_id)
        book_in_cart = False
        for cart_item in cart_items:
            if (
                cart_item.category_id == category_id
                and cart_item.product_id == product_id
            ):
                cart_item.augment_quantity(quantity-quantity)
                book_in_cart = True
                break

        if not book_in_cart:
            serializer.save()

        return Response(
            data={"message": "Add item to cart successfully."},
            status=status.HTTP_200_OK,
            content_type="application/json",
        )
    else:
        return Response(serializer.errors, status=400)

```